Beyond Simulation: Computer Aided Control System Design Using Equation-Based Object Oriented Modelling for the Next Decade

Francesco Casella[1] Filippo Donida[1] Marco Lovera[1]

[1]Politecnico di Milano, Dipartimento di Elettronica e Informazione, Italy
{casella,donida,lovera}@elet.polimi.it

Abstract

After 20 years since their birth, equation-oriented and object-oriented modelling techniques and tools are now mature, as far as solving simulation problems is concerned. Conversely, there is still much to be done in order to provide more direct support for the design of advanced, model-based control systems, starting from object-oriented plant models. Following a brief review of the current state of the art in this field, the paper presents some proposals for future developments: open model exchange formats, automatic model-order reduction techniques, automatic derivation of simplified transfer functions, automatic derivation of LFT models, automatic generation of inverse models for robotic systems, and support for nonlinear model predictive control.

Keywords Control system design, symbolic manipulation, model order reduction, CACSD.

1. Introduction

Control system engineering requires to master the dynamics of plants which are in general complex, interacting, multi-physics and multi-disciplinary. This explains why object-oriented modelling (OOM) and a-causal, equation-based, object-oriented languages (EOOL) always had a very strong connection with control system design. It is by no means accidental that much pioneering work in the OOM field was carried out within systems and control departments and research groups: consider, for example, the Omola language and the associated OmSim simulation environment, developed at the Department of Automatic Control of Lund Technical University [29, 30, 4], or the MOSES environment developed at the Dipartimento di Elettronica of Politecnico di Milano [26, 9]. During the '90, OOM was considered a very promising tool for Computer Aided Control System Design (CACSD), and there was a lot of activity in this field, which eventually culminated in the development of the Modelica Language [32].

At the beginning of that decade, papers appeared on the subject in the IEEE's Control Systems Magazine [31, 10], which discussed the potential of OOM for control system design. Reading those papers in retrospect shows that some of the promises where actually met or even exceeded: OOM is now a mature field, both from a theoretical side and from the point of view of available simulation tools. On the other hand, much work still has to be done on two fronts. The first one, which has a more "political" nature, is spreading the OOM culture among in the control engineering community, which is still largely dominated by block-oriented modelling, and by the (mis)use of Matlab/Simulink for physical systems modelling; this challenge is of paramount importance, but it out of the scope of this paper. The second one, instead, is to develop tools which allow to use EOOL models and tools not only for simulation, but also for the design of advanced control systems. The availability of such tools is crucial in order to narrow the gap between the large body of highly sophisticated control theory developed during the last 20 years, and the application of this theory to real-life cases, beyond textbook-sized examples. This is the topic of the present paper.

Given the background and the past experience of the authors, the discussion might be biased towards the Modelica language and related tools. However, strictly object-oriented features such as inheritance, encapsulation and hierarchical composition do not play any significant role in the analysis and proposals made within this paper, which essentially focuses on transformations of flattened models. On the contrary, the discussion is relevant for any equation-based modelling language, provided that it is a-causal and it allows symbolic manipulation of the equations by the compiler.

The paper is structured as follows: Section 2 gives a high-level view of the modelling activities required for control system design, while the following Section 3 discusses how currently available tools can help the control engineer in his/her task, with particular reference to Modelica tools. Sections 4 and 5, which are the core of the paper, propose several research and development directions to substantially increase the level of support to the control en-

2nd International Workshop on Equation-Based Object-Oriented Languages and Tools. July 8, 2008, Paphos, Cyprus.

gineer, willing to apply advanced control theory to real-life problems. Section 6 concludes the paper with final remarks.

2. The role of mathematical models in control system design

The design of control systems always requires some knowledge about the dynamic behaviour of the plant under control. When the plant design is mature and well-known, and the control system design is based on Proportional-Integral-Derivative (PID) controllers, the latter is often based on past experience and possibly on some empirical measurements. In this case, which covers the vast majority of installed industrial controllers, no (explicit) dynamical modelling is needed.

On the other hand, in an increasing number of cases, the performance of the control system is becoming a key competitive factor for the success of innovative, high-tech systems. To name a few examples, consider high-performance mechatronic systems (such as robots), vehicles enhanced by active integrated stability, suspension, and braking control, aerospace system, advanced energy conversion systems. All these cases possess at least one of the following features, which call for some kind of mathematical modelling for the design of the control system:

- closed-loop performance critically depends on the dynamic behaviour, which is not well-known in advance;
- the system is complex, made of many closely interacting subsystems, so that the behaviour of the whole system is more than just the sum of its parts;
- advanced control systems are required to obtain competitive performance, and these in turn depend on explicit mathematical models for their design;
- the system is very expensive and/or safety critical, requiring extensive validation of good control performance by simulation.

In most of these cases, two different classes of mathematical models are derived: compact models for control design and detailed models for system simulation.

2.1 Compact models for control design

Models belonging to this class are directly used for controller design, and are usually formulated in state-space form:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), p, t) \\ y(t) &= g(x(t), u(t), p, t) \end{aligned} \tag{1}$$

where x is the vector of state variables, u is the vector of system inputs (control variables and disturbances), y is the vector of system outputs, p is the vector of parameters, and t is the continuous time. A special case is that of linear, time-invariant models (LTI), which can be described as:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2}$$

or, equivalently, as a transfer function:

$$G(s) = C(sI - A)^{-1}B + D. \tag{3}$$

In many cases, the dynamics of systems in the form (1) is approximated by (2) via linearization around some equilibrium point. There is also a vast body of advanced control techniques which are based on discrete-time models:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), p, k) \\ y(k) &= g(x(k), u(k), p, k) \end{aligned} \tag{4}$$

where the integer time step k usually corresponds to the sampling time T_s of a digital control system. Many techniques are available to transform (1) into (4).

These models must capture the fundamental dynamics which is relevant for control system performance, while remaining as simple as possible: most advanced control design techniques start to become intractable for systems of order greater than about ten. If the models are simple enough, it is also sometimes possible to express the dependence of key dynamic features (such as, e.g., the natural frequency and damping coefficient of an oscillating dynamics) from plant design data. This can be very important to assess the impact of physical system design decisions on controller performance. For example, if the natural frequency of the first mode of oscillation limits the controller bandwidth, and it is found that this frequency mainly depends on the stiffness of a certain mechanical component, then it might be reasonable to change the mechanical design of that component in order to improve the overall performance.

In order to derive such simple models, it is usually necessary to introduce many, sometimes drastic, simplifying assumptions: all those phenomena that only marginally affect the equilibrium values and/or the control-relevant dynamics of the system are neglected. This activity requires highly skilled and experienced modellers, with a good knowledge of control design techniques, as well as of domain-specific strategies for model simplification.

2.2 Detailed models for system simulation

At the other end of the modelling spectrum, detailed simulation models can be found. Although it is always necessary to make reasonable modelling assumptions (a model is always a focused and limited description of the physical world), simulation models can include a lot more detail and second-order effects, since modern CPUs and simulation environments can easily handle complex systems with (tens of) thousands of variables. It is well-known that OOM methodologies and EOOLs provide very good support for the development of such models, thanks to equation-based modelling, a-causal physical ports, aggregation and inheritance. If the OOM model does not contain discrete variables and events, then it is basically equivalent to the set of DAEs:

$$F(x(t), \dot{x}(t), u(t), y(t), p, t) = 0 \tag{5}$$

Many EOOLs and tools also allow to describe hybrid systems, with discrete variables, conditional equations or expressions, and events. For example, see [7, 8] and references therein for hybrid system descriptions based on hybrid automata, or the Modelica language specification [41], in particular Appendix C. Although hybrid system control

is an interesting and emerging field, for the sake of conciseness this paper will focus on purely continuous-time physical models, with application to the design of continuous-time or sampled-time control systems.

These larger, more detailed models play a double role, with respect to those described in the previous sub-section. On one hand they allow to check how good (or crude) the compact models is, compared to a more detailed description, thus helping to develop good compact models. On the other hand, they allow to check the closed-loop performance of the controlled system, once a controller design is available. It is in fact well-known that validating the closed-loop performance using the same simplified model that was used for control system design is not a sound practice; conversely, validation performed with a more detailed model is usually deemed a good indicator of the control system performance, whenever experimental validation is not possible for some reason.

3. Overview of current CACSD practice with EOOLs

As of today, the practising control engineer already gets much support from EOOL-based tools for his/her control system design activities.

3.1 Support to control system synthesis

A typical starting point for the design of the control system is the analysis of the linearized dynamics of the plant, around one (or more) steady-state operating conditions. If the EOOL tool only supports simulation, then one can run open-loop simulations of the plant model, subject to step or to, e.g., pseudo-random binary sequence inputs, and then reconstruct the dynamics by system identification procedures.

A more direct approach, supported by many tools, is to directly compute the A, B, C, D matrices of the linearized system around specified equilibrium points, using symbolic and/or numerical techniques. The result is usually a high-order linear system, which can then be reduced to a low-order system by standard techniques for linear model order reduction, such as, e.g., balanced truncation.

A non-trivial issue with both approaches is the computation of the equilibrium point (what is sometimes called DC analysis in the field of electrical circuit simulation). In a typical setting, the desired steady-state values of the outputs $\bar{y}$ are known, and the tool must solve the steady-state initialization problem for the system (5):

$$F(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) = 0 \tag{6}$$

in order to find out the corresponding equilibrium values of the inputs $\bar{u}$ and of the states $\bar{x}$. This problem can be numerically challenging, because it often requires solving large systems of coupled nonlinear equations by iterative methods, which might fail if the iteration variables are not properly initialized. Currently available OOM tools (and, in particular, Modelica tools) are still far from providing general robust solutions to this problem. A sub-optimal approach to find equilibrium points is to initialize system (5) by giving tentative initial values to the state variables (which makes the initialization problem easier to solve) and then to simulate it until it reaches a steady state. If the system is asymptotically stable and the inputs $\bar{u}$ are known, this is relatively straightforward; otherwise, it is necessary to add suitable feedback controllers to drive the outputs to the desired values $\bar{y}$ and/or to stabilize the system. In both cases, the simulation of this initialization transient might fail for numerical reasons before reaching the steady state, due to a bad choice of the initial states.

3.2 Closed-loop performance assessment by simulation

Regardless of the actual design methodology, once the controller has been set up, an OOM tool can be used to run closed-loop simulations, including both the plant and the controller model. Many OOM tools provide model export facilities, which allow to connect an OO plant model with only causal external connectors (actuator inputs and sensor outputs) to a causal controller model in a causal simulation environment. From a mathematical point of view, this corresponds to reformulating (5) in state space form (1), by means of analytical and/or numerical transformations.

3.3 Development of simplified models

The object-oriented approach, and in particular replaceable components, allows to define and manage families of models of the same plant with different levels of complexity, by providing more or less detailed implementations of the same abstract interfaces. For example, consider a heat exchanger model: the abstract interface has four fluid connectors, two for the hot fluid inlet and outlet, and two for the cold fluid inlet and outlet. The corresponding implementation might range from a very simple static model based on log-mean temperatures, with a few algebraic equations, up to a very detailed finite volume model using nonlinear fluid properties and empirical correlations for heat transfer, and with dozens of state variables and a few hundred algebraic equations.

This feature of OOM allows to develop simulation models with different degrees of detail (and CPU load) throughout the entire life of an engineering project, from preliminary design down to commissioning and personnel training, within a coherent framework. However, this activity is based on manual work by the modeller, who needs to develop the different implementations explicitly. Moreover, it is often not easy to obtain compact models such as (1), because this requires to apply simplifications that may not fit well the abstract component boundaries.

3.4 Generation of real-time simulation code

An important step in the development of embedded control systems is Hardware-In-the-Loop simulation (HIL), where the real control hardware is tested by connecting it to a real-time simulator, instead of the real plant. Many currently available EOOL-based tools support automatic generation of efficient real time code starting from fairly large simulation models in the form (5). A common strategy for this purpose is to apply inline integration [12, 11] to (5), i.e. to

substitute the derivatives with their approximation formulae (e.g. Euler's formula), and then solve the system using all available numerical and symbolic techniques.

In order to provide real-time code which is fast enough, it is usually important to reduce the model complexity with respect to off-line simulation models - this can be done by following the approach sketched in Section 3.3.

3.5 Optimization

Some EOOL and tools support some kind of optimization, which might be useful for control system design. For example, the gPROMS language [6] has allowed to declare mixed-integer nonlinear optimization problems for a long time. More recently, extensions to the Modelica language were proposed to formulate optimization problems [2].

3.6 Future perspectives

It is the authors' view that EOOL-based tools should support advanced control system design problems in a much more direct way, by making extensive use of control-oriented symbolic manipulation techniques. Ideally, it would be good if the control engineer could develop a detailed simulation model by using object oriented tools and re-usable model libraries, then automatically obtain simplified, compact models which are already formulated as required by the specific control technique. The availability of such tools might promote the application of advanced, model-based techniques that are currently limited by the model development process.

Being aware that this is a very long-term goal, which might even require some kind of artificial intelligence, some first steps in this direction are discussed in the following sections, with particular reference to the Modelica language and Modelica compliant tools.

4. Basic enabling technologies

The advanced, control-oriented features of future EOOL tools need some basic enabling technologies and methodologies to build upon. These are briefly discussed in this section.

4.1 Open standards for model and data exchange

Advanced applications of OOM to control system design will most likely require to use different specialized tools in a coordinated fashion, rather than relying on one-fits-all comprehensive software tools. In fact, during the last decades, the number and the quality of simulation, design and analysis tools has increased enormously: there is plenty of open and closed source software for the simulation of physical systems, control synthesis, data analysis, test, validation, personnel training via a graphical user interface, etc. Some of these tools are developed for specific purposes, while other are more general in scope (e.g., symbolic manipulation tools, differential equation solvers, data analysis packages). Unfortunately, all this software development activity did not follow any standardisation process, leading to a great diversity in the representation of the information. The definition of standard interfaces will be useful for the information exchange between different applications; as a consequence, by providing a representation for all the stages of the model manipulation (starting from the translation, going to the flattening, to the model order reduction and so forth) it will be possible to make all the applications interact at different levels, thus combining positive effects from different applications and obtaining better results.

Exchange formats for model equations and for simulation data should probably be based on the XML language, for several reasons:

- the tree structure of XML documents easily allows to represent complex data structures, including symbolic representations of equations;
- XML documents can be read with standard text-editors and browsers, thus avoiding all the problem usually raised by obscure, ad-hoc binary formats;
- there exists a large base of software (open source and commercial) for the handling of XML files;
- by re-using this existing software, it is quite straightforward to translate an XML document representing a mathematical model into any other equation-based language, and vice-versa;
- binary XML formats can be used to reduce the verbosity of XML documents and the cost of parsing them;
- there exist some languages (e.g. DTD and XSD) to formally specify the structure of the information the XML file must contain.

Such standard interfaces for flattened Modelica models and their corresponding simulation data are currently being investigated at Politecnico di Milano using the OpenModelica compiler [16, 1] as a host EOOL environment, and symbolic manipulation tools such as Mathematica, Maple or Maxima as target environments. If the model is purely continuous-time, i. e., it is equivalent to the DAE (5), then MathML [42] on one side, and ModelicaXML [35] on the other side might constitute good starting points. If hybrid models are considered, one may consider all the languages developed for the description of hybrid automata in recent years [8], even though the class of hybrid systems which can be described in Modelica with *when* statements is larger than just hybrid automata.

4.2 Model Order Reduction

Another key enabling technology is represented by mixed numerical-symbolic Model Order Reduction (MOR) techniques. These have already been successfully applied to the analysis of electrical circuit models, which are based on DAE models such as (5), see [40, 17], and are currently available in commercial tools such as Analog Insydes [13]. The MOR strategies are based on the clever application of three fundamental steps:

- specify an allowed error bound, e.g. in terms of percentage error of certain steady-state output values corresponding to given constant inputs, or in terms of maximum deviation of some outputs from a reference trajectory obtained with given input signals, or in terms

of maximum error of small-signal frequency responses around a certain operating point and within a given frequency interval;

- derive a ranking of all terms in all equations, expressing how much each term has a significant effect on the required modelling accuracy;
- remove all terms in ascending order, until the specified error bound is reached.

Other MOR techniques exist to reduce large linear systems, based on concepts such as modal analysis and projection methods; see [38] for a comprehensive overview.

The application of such MOR tools and techniques, possibly with extended functionality and algorithms, looks very promising not only for the simplification of electrical circuit models, but also for the order reduction of generic, nonlinear DAE models, obtained from the flattening of generic EOOL models. This kind of techniques should allow to automatically obtain approximated compact models such as (1), starting from much more detailed simulation models, by formulating specific approximation bounds in control-relevant terms (e.g., percentage errors of steady-state output values, norm-bounded additive or multiplicative errors of weighted transfer functions, or ℓ_∞-norm errors of output transients in response to specified input signals). Given the ever-increasing computation power that can be expected by Moore's law, the future of these techniques for CACSD applications definitely appears bright.

4.3 Reliable steady-state initialization and static model inversion

A reliable support to the control engineer's activity requires to improve the techniques to solve the steady-state equations (6), which are usually the starting point for any kind of analysis, including MOR. As pointed out earlier, solving (6) requires iterative methods which might fail if not properly initialized. Troubleshooting can be very frustrating and time-consuming, and calls for experts of both simulation methods and domain-specific models. This is not acceptable in the envisioned framework, which is based on *automatic* manipulation by EOOL tools.

One option, which is currently being investigated at Politecnico, is to introduce extensions to the Modelica language to support homotopy methods, in a way similar to the approach followed by the SPICE circuit simulation program. The basic idea is that each model has two versions: the "easy" one, for which it is easier to find a steady-state solutions, and the "true" version, which is the model to be actually used for simulation. The two models share the same variables, but use different equations. The system model obtained by the aggregation of the "easy" models is represented by

$$F_e(x, \dot{x}, y, u, p, t) = 0, \tag{7}$$

while the aggregation of the "true" models leads to

$$F_t(x, \dot{x}, y, u, p, t) = 0, \tag{8}$$

The idea is now to first solve the initialization problem for (7), which should not give rise to significant numerical problems. The solution to this simplified problem constitutes the first guess for a new problem:

$$(1-\alpha)F_e(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) + \alpha F_t(\bar{x}, 0, \bar{u}, \bar{y}, p, 0) = 0, \tag{9}$$

which will be solved by varying α from 0 to 1 in small steps, eventually finding the steady-state solution of system (8). In general, this approach should help to reduce (and hopefully eliminate) the need to manually set initial guess values for iteration variables of initialization problems.

5. New functionalities for control system design

5.1 Simplified symbolic transfer functions

In many interesting cases, the performance of the control system is limited by the dynamic behaviour of the controlled plant. For example, poorly damped oscillations can limit the bandwidth of motion control systems, as well as non-minimum phase behaviour. The control engineer can gain a lot of useful insight from approximated transfer functions, where the dependence of the critical dynamic features from a few physical parameters is clearly visible. For instance, the natural frequency of a pair of complex poles in a mechanical system might depend mainly on the stiffness and on the mass of a certain physical component, or, the time constant of a right-half-plan zero in a fluid system might depend on the fluid velocity in a certain point.

This is a first case where automatic MOR techniques could prove extremely useful. Ideally, the user should specify the steady-state operating point, the relevant inputs and outputs, and some frequency-weighted error bounds, and get low-order approximated transfer functions of the linearized system, with approximated but explicit dependence of the transfer function features (gains, poles and zeros) from the physical model parameters. A suitable combination of EOOL tools (equipped with model import/export interfaces) with existing MOR tools like Analog Insydes [13] could provide very interesting results in this direction without too much effort.

5.2 Automatic derivation of LFT models

Once a model has been reduced to a low-order state-space form by the combined application of symbolic MOR techniques and clever model simplifications as explained in Section 3.3, it might be useful to automatically bring them in the form required for advanced control system design, using symbolic manipulation tools. Modern control theory provides methods and tools in order to deal with design problems in which stability and performance have to be guaranteed also in the presence of model uncertainty, both for regulation around a specified operating point and for gain scheduled control system design.

Most of the existing control design literature assumes that the plant model is given in the form of a Linear Fractional Transformation (LFT) (see, e.g., [46, 27]), a modelling paradigm which is currently an active research topic in the control engineering and system identification communities. In the robust control framework LFT models consist of a feedback interconnection between a *nominal* LTI

plant and a (usually norm-bounded) operator which represents model uncertainties, e. g., poorly known or time-varying parameters, nonlinearities, etc. A generic such LFT interconnection is depicted in Figure 1, where the nominal plant is denoted with P and the uncertainty block is denoted with Δ. Note that this representation is extremely general, and by no means limited to uncertain LTI systems; in fact, it is possible to describe any nonlinear DAE system by putting all the nonlinear functions in the Δ block and by providing an LTI model with direct feedthrough terms to describe the algebraic equations.

LFT models can be used for the design of robust and gain scheduling controllers, but they can also serve as a basis for structured model identification techniques, where the uncertain parameters that appear in the feedback blocks are estimated based on input/output data sequences.

The process of extracting uncertain parameters from the design model of the system to be controlled is a highly complex one. Symbolic techniques play a very important role in this process: the main use for such techniques is to find, via suitable pre-processing steps, equivalent representations of rationally dependent parametric matrices, which automatically lead to lower-order LFT representations. Tools already exist to perform this task [27].

The LFT modelling problem in its simplest form is associated with the problem of designing a controller for operation near a nominal operating point for the system. The problem is then formulated on a local linearised representation of the plant to be controlled and is familiarly termed "pulling out the Δs", i.e., it consists of manually or symbolically manipulating the linearised equations in order to separate the nominal part of the plant from the uncertain one, arranging them in a suitable feedback interconnection. This reformulation of the plant model lies at the vary basis of modern robust control theory and is currently supported by a number of different symbolic manipulation tools. A recent overview of the state-of-the-art in this research area can be found in [18]. As an example, consider a time-invariant, nonlinear state-space system in the form

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), p) \\ y(t) &= g(x(t), u(t), p), \end{aligned} \tag{10}$$

where p denotes a vector of uncertain parameters, and assume that the equilibrium condition $\bar{x}$, $\bar{u}$, $\bar{y}$, which solves the steady-state equations

$$\begin{aligned} 0 &= f(\bar{x}, \bar{u}, p) \\ \bar{y} &= g(\bar{x}, \bar{u}, p) \end{aligned} \tag{11}$$

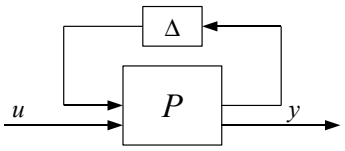

Figure 1. Block diagram of the typical LFT interconnection adopted in the robust control framework.

is available. Defining now the *deviation variables*

$$\delta x(t) = x(t) - \bar{x} \tag{12}$$

$$\delta u(t) = u(t) - \bar{u} \tag{13}$$

$$\delta y(t) = y(t) - \bar{y}, \tag{14}$$

it is possible to approximate the dynamics of (10) with a the following linear, parameter-dependent system

$$\begin{aligned} \dot{\delta x}(t) &= A(p)\delta x + B(p)\delta u \\ \delta y(t) &= C(p)\delta x + D(p)\delta u \end{aligned}, \tag{15}$$

where the four matrices A, B, C, D are the Jacobians of the two functions f and g:

$$A(p) = \frac{\partial f}{\partial x}, \quad B(p) = \frac{\partial f}{\partial u}$$
$$C(p) = \frac{\partial g}{\partial x}, \quad D(p) = \frac{\partial g}{\partial u}.$$

Under suitable assumptions (such as that the state space matrices are polynomial or rational functions of the elements of p, see, e.g., [46]) it is possible to transform the system description (15) into an LFT representation (see, again, Figure 1). As mentioned previously, converting (15) into an LFT with a Δ block of minimum dimension is a non-trivial symbolic manipulation problem.

An even more challenging formulation of the LFT modelling problem is the one of *simultaneously* representing in LFT form all the linearisations of interest for control purposes of the given nonlinear plant. Indeed, in many control engineering applications a single control system must be designed to guarantee the satisfactory closed-loop operation of a given plant in many different operating conditions in the presence of parametric and possibly non parametric uncertainty. The gain scheduling approach to the problem, which has been part of the engineering practice for decades, can be roughly summarised as follows: find one or more *scheduling variables* α which can completely parametrise the operating space of interest (e.g., the flight envelope in the case of aircraft control) for the system to be controlled; define a parametric *family* of linearised models for the plant associated with the set of operating points of interest; finally, design a *parametric* controller which can both ensure the desired control objectives in each operating point and an acceptable behaviour during (slow) transients between one operating condition and the other. Many design techniques are now available for this problem (see, e.g., [5, 22, 37]), which can be reliably solved, provided that a suitable model in parameter-dependent form has been derived for the system to be controlled. The goal here would be to arrive at a representation of the dynamics of the nonlinear system in the form depicted in Figure 2, which is usually denoted as an LPV-LFT system, where the LPV acronym stands for Linear Parameter-Varying. The model structure now includes two feedback interconnections: the block $\Delta(p)$ takes into account the presence of the uncertain parameter vector p, while the block $\Theta(\alpha)$ models the effect of the varying operating point, parametrised by the vector of time-varying parameters α.

The state-of-the-art of modelling for gain scheduling can be briefly summarised by defining two classes of modelling approaches: *analytical* methods based on the availability of (relatively) reliable nonlinear equations for the dynamics of the plant, from which suitable control-oriented representations can be derived (see, e.g., [28] and the references therein); *experimental* methods based entirely on identification, i.e., aiming at deriving LPV models for the plant directly from input/output data (see, among many others, [21, 45, 23]). The methods belonging to the first class aim at developing LPV models for the plant to be controlled by resorting to, broadly speaking, suitable extensions of the familiar notion of linearisation, developed in order to take into account off-equilibrium operation of the system. As far as experimental methods are concerned, most LPV identification techniques are based on the assumption that the identification procedure can rely on one *global* identification experiment in which both the control input and the scheduling variables are (persistently) excited in a simultaneous way. This assumption may not be a reasonable one in many applications, in which it would be desirable to try and derive a parameter-dependent model on the basis of *local* experiments only, i.e., experiments in which the scheduling variable is held constant and only the control input is excited. Such a viewpoint has been considered in [43, 34, 23], where numerical procedures for the construction of parametric models for gain scheduling on the basis of local experiments and for the interpolation of local controllers have been proposed.

To our best knowledge the only documented attempt at deriving control-oriented LFT models automatically from a nonlinear simulator is presented in [44], where the focus was on the automatic generation of LFT models for aerospace applications. Much remains to be done. An EOOL-based CACSD tool dealing with the generation of control-oriented LFT models should allow to specify some error bounds for the system approximation (with respect to steady-state, transient, and frequency response), the choice of input, output and scheduling variables, and the choice of parameters to include in the LFT representation. Based on that, it should be able to automatically compute the structure of the interconnections defined in Figures 1 and 2 for the robust and gain-scheduling control design problems, respectively, the state-space matrices of the nominal part P of the model (either as analytical expressions, if possible, or at least as algorithms for their computation) and analytical or algorithmic representations of the feedback blocks $\Theta(\alpha)$ and $\Delta(p)$. Finally, it is apparent from the short literature review presented above that currently only physical and black-box modelling methods are available, while no general purpose CACSD tools capable of combining first principles models and experimental data in a single control-oriented model seem to exist. The convergence of the two modelling approaches both in terms of methods and tools would be a very desirable outcome of the research in this field.

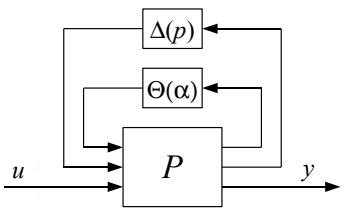

Figure 2. Block diagram of the typical LFT interconnection adopted in the robust LPV control framework.

5.3 Automatic computation of inverse models for robotic systems

The design of controllers for non-redundant robotic manipulators with N degrees of freedom usually starts from the equations of motion obtained from the Euler-Lagrange equations [39]:

$$B(q)\ddot{q} + H(q,\dot{q})\dot{q} + g(q) = \tau \tag{16}$$

$$y_p = K(q) \tag{17}$$

$$y_v = \frac{\partial K}{\partial q}\dot{q}, \tag{18}$$

where q is the N-element vector of Lagrangian coordinates, which usually correspond to the rotation angles of the actuator motors, $\dot{q}$ is the vector of the corresponding generalized velocities, y_p describes the position and orientation vector of the end effector, y_v contains the corresponding generalized velocities, τ is the vector of generalized applied forces corresponding to each degree of freedom (usually the torques applied by rotating actuators), $B(q)$ is the inertia matrix, $H(q,\dot{q})$ is the matrix corresponding to the centripetal, Coriolis, and viscous friction forces, while the vector $g(q)$ accounts for the effects of the gravitational field; all vectors have dimension N.

The classical approach to write (16) requires to compute the so-called direct kinematics (DK), i.e. how the values of q and $\dot{q}$ translate into the position and motion of the robot's end effector, then to compute the Lagrange function, i.e. the difference between kinetic and potential energy, and apply the Euler-Lagrange equations. This can be done manually, or using one of the specialized tools available for this task. Equations (16)-(18) can then be used as a basis for both controller design and system simulation.

Within an OOM approach, it is possible to save much time by developing an object-oriented model using an EOOL, e.g. using the Modelica MultiBody library [33]. Due to the kinematic constraints imposed by the joints, the original flattened model corresponds to an index-3 DAE,

$$F(x,\dot{x},y,u) = 0, \tag{19}$$

which is mathematically equivalent to the Lagrange model (16)-(18).

Currently available Modelica tools tackle the problem by applying specialized algorithms, which exploit the knowledge of the topology of the kinematic chain, as well as standard techniques such as BLT partitioning, tearing, dummy derivatives and symbolic solution of equations

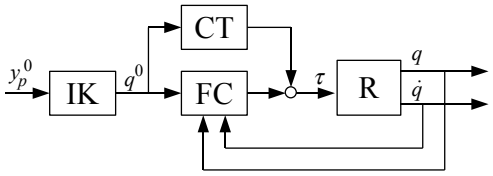

Figure 3. Block diagram of computed torque control

[33]. From a conceptual point of view, a change of state variables x allows to transform (19) into an index-1 system

$$F_1(x, \dot{x}, y, u) = 0, \tag{20}$$

where

$$x = \begin{bmatrix} x_p \\ x_v \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad y = \begin{bmatrix} y_p \\ y_v \end{bmatrix}, \quad u = \tau. \tag{21}$$

Eventually, efficient procedures are produced to solve (20) for $\dot{x}$ and y given x and u, thus actually bringing the system into state-space form:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= g(x, u). \end{aligned} \tag{22}$$

This formulation can be used to solve simulation problems, by linking it to any ODE/DAE solver. However, there are several other interesting things that could be done with (20), from a control engineer's perspective.

Robot trajectories are originally defined in terms of end effector coordinates as functions of time $y_p^0(t)$. In order to obtain the corresponding reference trajectories in Lagrangian coordinates for the low-level robot joint controllers, (17)-(18) must be solved for q, $\dot{q}$, thus computing the so-called inverse kinematics (IK):

$$q^0 = K^{-1}(y_p^0) \tag{23}$$

$$\dot{q}^0 = \left(\frac{\partial K}{\partial q}\right)^{-1} y_v^0; \tag{24}$$

note that the Jacobian of $K(q)$ is also needed to solve (23), since analytical inverses cannot usually be obtained. Furthermore, two interesting approaches to model-based robot control are based on suitable manipulations of eq. (16): the *pre-computed torque* approach and the *inverse dynamics* approach [39].

The pre-computed torque approach is a feed-forward compensation scheme, where the theoretical torque required to follow the reference trajectory is directly fed to the torque actuators (see Fig. 3) in order to obtain a good dynamic response to the set point y_p^0. The CT block performs this task by solving (16) for τ, given the reference trajectory and its derivatives:

$$\tau = B(q^0)\ddot{q}^0 + H(q^0, \dot{q}^0)\dot{q}^0 + g(q^0). \tag{25}$$

A feedback controller (FC) is also included to deal with uncertainties and disturbances.

The inverse dynamics approach is a feedback compensation scheme, that uses the model in order to transform

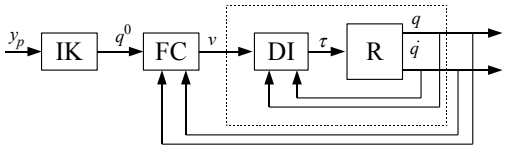

Figure 4. Block diagram of inverse dynamics control

the non-linear control problem into a linear, time-invariant one. Define a virtual input variable v, which satisfies the following equation

$$\tau = B(q)v + H(q, \dot{q})\dot{q} + g(q). \tag{26}$$

Since the inertia matrix B is structurally non-singular, it is always possible to solve (26) for v:

$$v = B^{-1}(q)\left(\tau - H(q, \dot{q})\dot{q} - g(q)\right). \tag{27}$$

Plugging v in the robot dynamics equation (16), one obtains:

$$\ddot{q} = v. \tag{28}$$

The block diagram interpretation of these equations is shown in Fig 4: thanks to the dynamic inversion (DI) block, the dynamic relationship between the virtual input v and the Lagrangian positions and velocities q and $\dot{q}$ (represented by the dotted block) is now described by a simple integrator and a double integrator, respectively. It is then easy to tune a fixed-parameter, linear feedback controller (FC) in order to obtain the desired closed-loop dynamics.

Starting from the index-1 DAE robot model (20), it is straightforward to derive the equations and then the explicit algorithms to compute the DK, IK, CT, and DI, by using the same techniques employed to bring (20) into state-space form. The DK (17)-(18) is obtained by solving (20) for y_p (and possibly y_v) given q (and possibly $\dot{q}$), while the IK is obtained by solving (20) for q (and possibly $\dot{q}$) given y_p (and possibly y_v); the subset of required equations is found by suitable analysis of the incidence matrix. The CT (25) is obtained by solving (20) for τ given q, $\dot{q}$, and $\ddot{q}$. Finally, the DI (26) is obtained by solving (20) augmented with (26) for τ given v, q, and $\dot{q}$. EOOL tools should then be able to automatically generate the code corresponding to the DK, IK, CT, and DI blocks in two forms: as algorithms to compute the outputs given the inputs (e.g., C code for direct inclusion in the robot controller), as well as equation-based Modelica blocks, which could be used for closed-loop simulation within a Modelica environment.

As a final remark, note that the method of inverse dynamics is a special case of the much more general theory of feedback linearization [20], whose goal is to obtain a LTI dynamics made by pure integrators from generic nonlinear systems, by applying suitable feedback actions as shown in Figure 4. It could also be interesting to investigate the coupling between EOOL tools and symbolic manipulation tools for the design of such controllers.

5.4 Fast and compact models for Model Predictive Control

The Model Predictive Control (MPC) approach [25, 36] is based on a few key ideas, that turn the control problem into an optimization problem. The control variable is a discrete-time variable, that changes periodically every T_s seconds:

$$u(t) = u(k), \quad kT_s \leq t < (k+1)T_s. \tag{29}$$

At each time step k, an optimization problem is solved, whose unknowns are the next values of the control variable $u(k+i)$ over a finite horizon $1 \leq i \leq N$. The first sample $u(k+1)$ is then applied to the actuators at the next time step, the rest of the values are discarded, and the process is repeated over and over, thus implementing a *receding horizon* strategy.

There are different ways to formulate the MPC problem, depending on the specific technique used to solve the problem. Generally speaking, the figure of merit to be minimized is a quadratic function, which suitably weights the future deviations of the controlled variables from the set point and the intensity of the control action, as well as any other problem-specific performance index that has to be minimized, e.g. the financial cost of running the process. The constraints of the optimization problem are the dynamic relationship between the input and output variables, typically in the form (4), and possibly other constraints, such as upper and lower bounds of the state, control and output variables and of their rate of change.

The main advantage of MPC is its intrinsic ability to deal with highly interacting multivariable systems (many control inputs and controlled outputs), while keeping into account operating constraints such as actuator saturations or hard bounds on controlled variables, and at the same time meeting some problem-specific optimality criterion. The main drawback is the high computational load, since a (possibly non-linear and non-convex) constrained optimization problem must be solved at each sampling time; this makes MPC suitable for systems with slow dynamics, e. g. chemical plants, where there is plenty of time to carry out the required computations in real time. This limitation is likely to become less and less stringent in the future, thanks to Moore's law.

The second issue is the requirement that a suitable plant model is available, as the control system performance critically depends on the model quality. Models for linear MPC can be obtained either by linearization of analytical models, or by system identification from experimental and/or simulation data, e. g. step responses; both cases are already supported by current EOOL tools. Nonlinear MPC (NMPC) algorithms are preferably based upon analytical models in state-space form (1), which are derived from physical first-principles models. The conversion to discrete-time form (4) is often performed internally by the NMPC algorithm itself, by standard ODE integration routines. This means that the interface between the EOOL tool and the NMPC tool is similar to the one used for simulation problems, i.e. the state-space form (1), possibly augmented by the Jacobians of the right-hand-sides of (1).

The main requirement for NMPC-oriented models is that they must have the least possible number of state and algebraic variables, in order to keep the complexity of the optimization problem within acceptable limits, and that they have good smoothness properties, in order to avoid convergence problems of the iterative optimization algorithms. The development of those models can be very time consuming, and require highly skilled manpower; it is apparent how better tool support could be extremely useful in order to reduce the development effort and cost.

The potential of OOM for MPC was first noted by Maciejowski at the end of the '90 [24]. There are several reported case studies [14, 15, 3, 19], where the model used in the NMPC algorithm was derived from a Modelica model of the physical plant, using the tool Dymola to produce the code corresponding to the state-space form (1), i.e., the *dsmodel.c* code that is usually linked to ODE/DAE solvers. In order to derive suitably simplified models, the features of Modelica discussed in Section 3.3 have been extensively exploited. In general, this approach has proven much more satisfactory than writing the C-code of the model from scratch; however, it still requires a substantial investment of time and effort for each new application.

The application of the automatic MOR techniques described in section 4, possibly still combined with some manual intervention in terms of replaceable models, looks very promising in order to bring detailed simulation models into a form which is suitable for NMPC with a much more limited effort by the developer.

Furthermore, [19] correctly points out that, although the interface to NMPC algorithms is very similar to the interface to ODE/DAE solvers, the former requires some more flexibility. For example, advanced NMPC schemes can provide on-line estimation of uncertain parameters through the use of extended or unscented Kalman filters. This means that some model parameters are no longer constant throughout a transient, so that the C-code obtained for simulation purposes must be manually adapted. A better option would be to implement a code export interface which makes it possible to turn selected parameters appearing in (5) (which are going to be estimated on-line) into inputs, before transforming the system in state-space form (1).

6. Conclusions

After a brief review of the different uses of models in control system design, the current state of the art of EOOL-based tools for CACSD has been reviewed: apparently, currently available tools mainly focus on simulation tasks. Several further directions for research and development in EOOL tools where then discussed, which go beyond the mere simulation problem. Results in these directions could substantially improve the level of support to the control engineer willing to apply advanced, model-based control techniques to real-life problems, starting from object-oriented models of the plant.

References

[1] OpenModelica home page. URL: `http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html`.

[2] J. Åkesson. Optimica - An extension of Modelica supporting dynamic optimization. In *Proceedings 6th International Modelica Conference*, pages 57–66, Bielefeld, Germany, Mar. 3–4 2008.

[3] J. Åkesson and O. Slätteke. Modeling, calibration and control of a paper machine dryer section. In *Proceedings 5th International Modelica Conference*, pages 411–420, Vienna, Austria, Sep. 4–5 2006.

[4] M. Andersson, S. E. Mattsson, D. Brück, and T. Schöntal. OmSim - an integrated environment for object-oriented modelling and simulation. In *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, CACSD'94*, pages 285–290, Tucson, Arizona, March 1994.

[5] P. Apkarian and R. J. Adams. Advanced Gain-Scheduling Techniques for Uncertain Systems. *IEEE Transactions on Control System Technology*, 6:21–32, 1998.

[6] P. I. Barton and C. C. Pantelides. The modeling of combined continuous and discrete processes. *AIChE Journal*, 40:966–979, 1994.

[7] D.A. van Beek, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Foundations of an interchange format for hybrid systems. In A. Bemporad, A. Bicchi, and G. Butazzo, editors, *Computation and Control, 10th International Workshop*, volume 4416 of *Lecture Notes in Computer Science*, pages 587–600. Springer Verlag, 2007.

[8] D.A. van Beek, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Concrete syntax and semantics of the compositional interchange format for hybrid systems. In *Proc. 17th IFAC World Congress*, Seoul, Korea, Jul 6–11 2008.

[9] E. Carpanzano and C. Maffezzoni. Symbolic manipulation techniques for model simplification in object-oriented modelling of large scale continuous systems. *Mathematics and Computers in Simulation*, 48(2):133–150, 1998.

[10] F. E. Cellier and H. Elmqvist. Automated formula manipulation supports object-oriented continuous-system modeling. *IEEE Control Systems Magazine*, 13(2):28–38, 1993.

[11] H. Elmqvist, S. E. Mattsson, and H. Olsson. New methods for hardware-in-the-loop simulation of stiff models. In *Proceedings 2nd International Modelica Conference*, pages 59–64, Oberpfaffenhofen, Germany, Mar. 18–19 2002.

[12] H. Elmqvist, M. Otter, and F. Cellier. Inline integration: A new mixed symbolic /numeric approach for solving differential–algebraic equation systems. In *Proc. ESM'95, European Simulation Multiconference*, pages xxiii–xxxiv, Prague, Czech Republic, Jun. 5–8 1995.

[13] The Fraunhofer-Institut für Techno-und Wirtschaftsmathematik. Analog Insydes. URL: `http://www.analog-insydes.de/`.

[14] R. Franke. Formulation of dynamic optimization problems using modelica and their efficient solution. In *Proceedings 2nd International Modelica Conference*, pages 315–323, Oberpfaffenhofen, Germany, Mar. 18-19 2002.

[15] R. Franke, M. Rode, and K. Krüger. On-line optimization of drum boiler startup. In *Proceedings 3rd International Modelica Conference*, pages 287–296, Nov. 3–4 2003.

[16] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nyström, L. Saldamli, D. Broman, and A. Sandholm. OpenModelica - A free open-source environment for system modeling, simulation, and teaching. In *Proceedings IEEE International Symposium on Computer-Aided Control Systems Design*, Munich, Germany, Oct. 4–6 2006.

[17] T. Halfmann and T. Wichmann. Symbolic methods in industrial analog circuit design. In *Scientific Computing in Electrical Engineering*, Mathematics in Industry. Springer Verlag, 2006.

[18] S. Hecker and A. Varga. Symbolic manipulation techniques for low order LFT-based parametric uncertainty modelling. *International Journal of Control*, 79(11):1485–1494, 2006.

[19] L. Imsland, P. Kittilsen, and T. S. Schei. Model-based optimizing control and estimation using modelica models. In *Proceedings 6th International Modelica Conference*, pages 301–310, Bielefeld, Germany, Mar. 3–4 2008.

[20] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002.

[21] L. Lee and K. Poolla. Identification of linear parameter-varying systems using nonlinear programming. *Journal of Dynamic Systems, Measurement and Control - Transactions of the ASME*, 121(1):71–78, 1999.

[22] D. J. Leith and W. E. Leithead. Survey of gain-scheduling analysis and design. *International Journal of Control*, 73(11):1001–1025, 2000.

[23] M. Lovera and G. Mercere. Identification for gain-scheduling: a balanced subspace approach. In *2007 American Control Conference*, New York, USA, 2007.

[24] J. M. Maciejowski. Modelling and predictive control: enabling technologies for reconfiguration. *Annual Reviews in Control*, 23(1):13–23, 1999.

[25] J. M. Maciejowski. *Predictive control: with constraints*. Prentice Hall, 2002.

[26] C. Maffezzoni and R. Girelli. Modular modelling in an object-oriented database. *Mathematical Modelling of Systems*, 4:121–147, 1998.

[27] J.-F. Magni. Linear fractional representation toolbox. Technical Report TR 6/08162 DCSD, ONERA, 2004.

[28] A. Marcos and G. Balas. Development of linear-parameter-varying models for aircraft. *Journal of Guidance, Control and Dynamics*, 27(2):218–228, 2004.

[29] S. E. Mattsson, M. Andersson, and K. J. Åström. Modeling and simulation of behavioral systems. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, volume 4, pages 3636–3641, San Antonio, Texas, Dec. 1993.

[30] S. E. Mattsson, M. Andersson, and K. J. Åström. Object-oriented modelling and simulation. In D. Linkens, editor, *CAD for Control Systems*, pages 31–69. Marcel Dekker Inc., New York, 1993.

[31] S. E. Mattsson and H. Elmqvist. Simulator for dynamical systems using graphics and equations for modeling. *IEEE Control Systems Magazine*, 9(1):53–58, Jan. 1989.

[32] S. E. Mattsson, H. Elmqvist, and M. Otter. Physical system

modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, 1998.

[33] M. Otter, H. Elmqvist, and S. E. Mattsson. The new Modelica MultiBody library. In *Proceedings 3rd International Modelica Conference*, pages 311–330, Linköping, Sweden, Nov. 3–4 2003. URL: `http://www.modelica.org/events/Conference2003/papers/h37_Otter_multibody.pdf`.

[34] B. Paijmans, W. Symens, H. Van Brussel, and J. Swevers. A gain-scheduling-control technique for mechatronic systems with position-dependent dynamics. In *Proceedings of the 2006 American Control Conference*, Minneapolis, USA, 2006.

[35] A. Pop and P. Fritzson. ModelicaXML: A Modelica XML representation with applications. In *Proceedings of the 3rd International Modelica Conference*, pages 419–430, Linköping, Nov 3–4 2003.

[36] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[37] W. Rugh and J. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

[38] P. Schwarz, J. Bastians, C. Clauss, J. Haase, A. Köhler, G. Otte, and P. Schneider. A tool-box approach to computer-aided generation of reduced-order models. In *Proceedings EUROSIM 2007*, 2007.

[39] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer Verlag, 2000.

[40] R. Sommer, T. Halfmann, and J. Broz. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. In *Proceedings of EUROSIM 2007*, Ljubljana, Slovenia, Sep 9–13 2007.

[41] The Modelica Association. Modelica - A unified object-oriented language for physical systems modeling - Language specification version 3.0. Online, Sep. 5 2007. URL: `http://www.modelica.org/news_items/documents/ModelicaSpec30.pdf`.

[42] The World Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0. Online, Oct 21 2003. URL: `http://www.w3.org/TR/MathML2/`.

[43] J.J.M. van Helvoort, M. Steinbuch, P.F. Lambrechts, and R. van de Molengraft. Analytical and experimental modelling for gain-scheduling of a double scara robot. In *Proceedings of the 3rd IFAC Symposium on Mechatronic Systems*, Sydney, Australia, 2004.

[44] A. Varga, G. Looye, D. Moormann, and G. Gräbel. Automated generation of LFT-based parametric uncertainty descriptions from generic aircraft models. *Mathematical and Computer Modelling of Dynamical Systems*, 4(4):249–274, 1988.

[45] V. Verdult. *Nonlinear system identification: a state space approach*. PhD thesis, University of Twente, 2002.

[46] K. Zhou, J. U. Doyle, and K. Glover. *Robust and optimal control*. Prentice-Hall, New Jersey, 1996.